NASA/TM-1999-209335

Configuration Management of an Optimization Application in a Research Environment

James C. Townsend, Andrea O. Salas, and M. Patricia Schuler
Langley Research Center, Hampton, Virginia

June 1999

NASA/TM-1999-209335

Configuration Management of an Optimization Application in a Research Environment

James C. Townsend, Andrea O. Salas, and M. Patricia Schuler
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

June 1999

INTRODUCTION

Multidisciplinary design of aerospace systems is a complex, computationally intensive process that combines system analyses (simulations) with design space search and decision making. The decision making is based on engineering judgement, but is greatly assisted by automation. Because the point of view, design emphasis, and design approach of specialists from various disciplines can be quite different, the practice has often been for each to optimize independently, with limited direct interaction or communication among disciplines. The present aim of multidisciplinary design optimization (MDO) is to meet the needs for increased interdisciplinary communication and reduced design cycle time.[1] The development of computational frameworks offers the capability to meet these needs via the use of sophisticated computational procedures combined with state-of-the-art optimization or design improvement techniques.

For several years, NASA Langley Research Center (LaRC) developed the Framework for Interdisciplinary Design Optimization (FIDO)[2] under the High Performance Computing and Communication Program (HPCCP). The goal of the FIDO project was to develop a framework for integrating high-fidelity engineering analyses to be executed in a distributed, heterogeneous computing environment. Increasingly complex multidisciplinary models of the High-Speed Civil Transport (HSCT, Fig. 1) were implemented in FIDO. In 1997, the LaRC HPCCP project began the transition to a much more comprehensive HSCT model, called HSCT4.0, involving high-fidelity analysis codes and a more realistic, highly complex analysis process. The FIDO approaches to integration and distributed computing are not being used to implement HSCT4.0. More recent framework technologies are being investigated as part of this project. Therefore, the HSCT4.0 application has been implemented in CORBA-Java Optimization (CJOpt), a new framework that uses Java with CORBA (Common Object Request Broker Architecture[3]) compliant software.

Like the preceding HSCT projects, HSCT4.0 is a research project involving the evolution of the application problem and the implementation strategy as both the MDO issues and CJOpt framework issues are better understood. Thus, in contrast to the software development ideal of fully specified requirements preceding design and development, the HSCT4.0 requirements are expected to change; therefore, development must occur in a flexible environment. Earlier versions of the HSCT were developed with simple, manual software configuration management (SCM) procedures that proved to be ineffective. The HSCT4.0 project's SCM and software engineering managers have developed an SCM plan with the goal of ensuring the integrity and traceability of changes made to the software, while maintaining a research environment. It is anticipated that, as experience is gained with the Plan, the procedures may be adjusted to allow further improvements for use within a research environment. This report describes how SCM is being applied to the HSCT4.0/CJOpt project. First, the HSCT4.0 analysis and CJOpt implementation are briefly discussed. Next, the general concepts of SCM and their implementation within the HSCT4.0/CJOpt project are described. The final section concludes with a discussion of the HSCT4.0/CJOpt experiences with SCM and some of the lessons learned from these experiences.

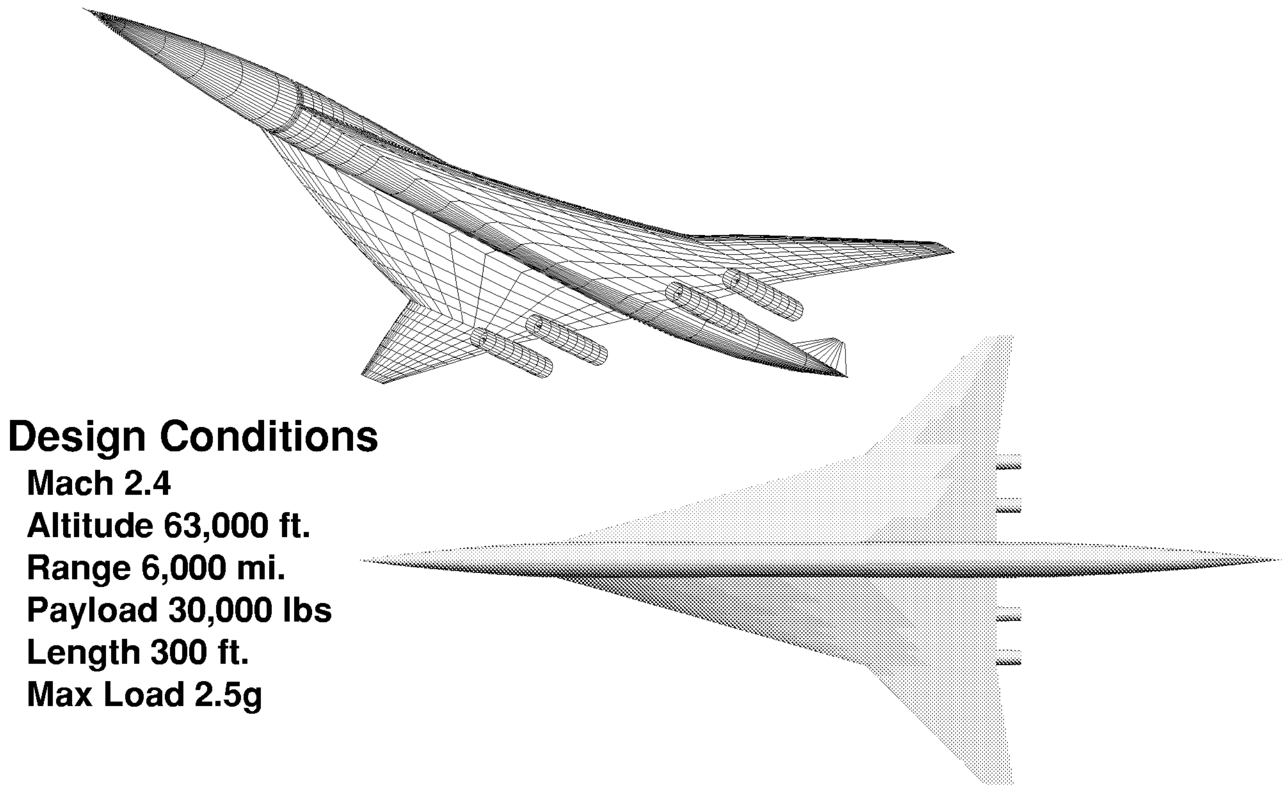


Figure 1. High-speed civil transport (HSCT) configuration implemented in FIDO.

HSCT4.0/CJOPT PROJECT

HSCT4.0/CJOpt System Background

The goal of the HSCT4.0/CJOpt project (as with HSCT/FIDO before it) is to investigate the use of a distributed, heterogeneous computing system to facilitate communications, apply computer automation, and introduce parallel computing to produce a true MDO process. The project has three purposes: to demonstrate technical feasibility, to demonstrate usefulness for selected applications, and to provide a working environment for use by LaRC researchers testing various optimization schemes. Philosophically, the system can be considered as a tool to be applied by a *design manager* who needs to improve the design of a complex product. Basically, HSCT4.0/CJOpt automates the coordination of analyses by the various disciplines (each on its assigned computer) into an overall optimization scheme.

HSCT4.0 Application

Increasingly complex multidisciplinary models of the HSCT were implemented in FIDO. The framework was first demonstrated for a version of this design problem with fast, limited-fidelity discipline codes (equivalent plate structural analysis, linearized aerodynamic analysis, propulsion table lookup, and a simple range equation for performance weight estimation), a geometry given by a set of points, a small number of design variables (on the order of ten), and a simple objective function. The most complex HSCT application implemented with FIDO demonstrated medium-fidelity structural (coarse-grain, finite-element analysis) and aerodynamic (marching supersonic Euler) codes coupled in a static aeroelastic loop.

The HSCT4.0 application (Fig. 2) provides the additional realism afforded by nonlinear corrections to linearized aerodynamic analyses, finite-element analysis on a realistic model that also contributes a structural weights estimation, full mission-cycle performance evaluation, and an actual proposed HSCT geometry with realistic constraints (such as disallowing ground scrape). This version of the HSCT problem has more than two hundred design variables and over thirty thousand constraints. At least five times as many processes as were used in the earlier HSCT analyses are involved in each design iteration.

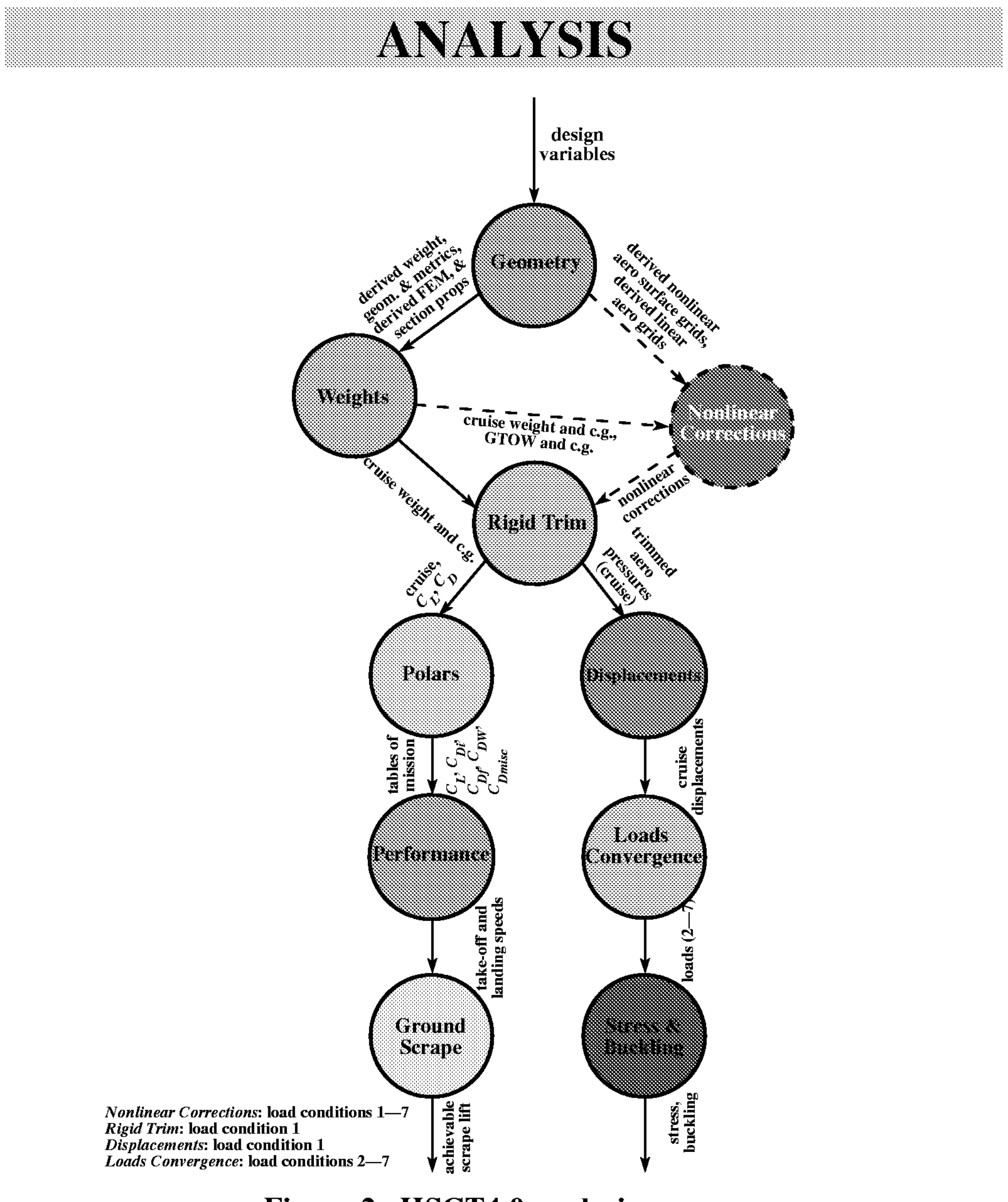

ANALYSIS
design variables
Geometry
derived weight, geom. & metrics, derived FEM, & section props
derived nonlinear aero surface grids, derived linear aero grids
Weights
cruise weight and c.g., GTOW and c.g.
Nonlinear Corrections
nonlinear corrections
cruise weight and c.g.
Rigid Trim
cruise, C_L, C_D
trimmed aero pressures (cruise)
Polars
Displacements
tables of mission C_L, C_{Df}, C_{DW}, C_{Dmisc}
cruise displacements
Performance
Loads Convergence
take-off and landing speeds
loads (2—7)
Ground Scrape
Stress & Buckling
achievable scrape lift
stress, buckling
Nonlinear Corrections: load conditions 1—7
Rigid Trim: load condition 1
Displacements: load condition 1
Loads Convergence: load conditions 2—7

Figure 2. HSCT4.0 analysis process

HSCT4.0/CJOpt Architectural Design

The HSCT4.0/CJOpt system design is object oriented. Whereas the distributed FIDO modules communicated through a special-purpose library, CJOpt uses a commercial implementation of the CORBA[3] specification to encapsulate system modules as executables in an object framework. Within this framework, CORBA acts as a *software bus* to connect objects across a network of computers in a client-server paradigm. The Object Request Broker (ORB) mediates the transfer of messages among the CORBA objects. The coding for encapsulating objects is written in the Java programming language. This allows the conversion of legacy code to Java modules (called *beans*), which promotes flexibility in the construction of the problem by using Java visual programming packages. Finally, a commercial relational database is used to facilitate data sharing among disciplines. A prototype CORBA-Java implementation of an early version of the HSCT application was compared to the FIDO implementation to verify feasibility.[4]

HSCT4.0 SOFTWARE CONFIGURATION MANAGEMENT

Defining the Plan

Software configuration management. SCM defines a set of methods and tools for identifying and controlling software during its development and use. Typical SCM activities include baseline establishment, change control and tracking, and reviews of the evolving software. The application of SCM increases the reliability and quality of software. Software configuration management is typically applied to the development of software applications, such as business, control systems, and engineering, with clear requirements and a well-defined life cycle. In research, it has been used for experimental applications in which the software plays a supporting role but is not part of the research. The HSCT4.0 application is the first at LaRC that has used such a tool for a purely research project. An earlier application of software engineering practices to research software at LaRC was the experimental use of the Software Engineering Evaluation System (SEES) as a method for independent verification and validation (IV&V) of MDO software.[5]

The expected benefits of an SCM process include consistent version control of each software item (code, test data, test procedure, or document), minimized risk of losing valuable information, clearly established roles and responsibilities, and assured ability to retrieve correct previous versions of software. Version control is particularly important, because it helps to ensure all developers use consistent software versions.

The simple, manual SCM methods that had been used for configuration management during the development of the previous HSCT problems were inadequate. Some of the difficulties encountered included losing track of changes to the codes, poor handling of changes required by operating system updates, keeping insufficient records of the reasons for changes, and applying version identifiers inconsistently to test and component codes. Consequently, additional work was required to reconstruct lost (or misplaced) versions when they were needed for testing new frameworks or communications systems.

SCM Plan. Because most HSCT4.0 project team members were inexperienced with formal SCM processes, during the early stages of the project, consultants were employed to recommend an SCM Plan. They were also tasked to conduct a pilot study with a few team members using one of the simpler HSCT/FIDO applications. The purpose of the pilot was to gain experience with the SCM software tools, refine SCM processes, and define how the software tools would support these processes. The experiences gained and lessons learned from the pilot were to be incorporated into training for the full project team. At the time, the CJOpt methodology had not been selected, and the team expected that the HSCT4.0 application might be developed as an extension of FIDO approach.

An SCM "needs analysis" was performed by the consultants, based on the following factors: software size (over 40 significant modules to be managed), staff size (approximately 15), complexity (multiple distributed disciplines, heterogeneous computer systems, parallelism, complex data flow), and criticality (high-level milestones to be met).[6] The HSCT4.0/CJOpt project's SCM and software engineering managers used the results of this analysis to develop the SCM Plan. The nature of the research environment in which the software is being developed was seriously considered while developing the Plan. In the research environment, requirements necessarily evolve as the research progresses. Therefore, the SCM Plan for HSCT4.0/CJOpt was made to be more flexible than typical plans (see, e.g., the Institute of Electrical and Electronics Engineers (IEEE) 828 Standard[7]), and it is anticipated that the Plan will have to be adjusted to accommodate research necessities as experience is gained with SCM.

The SCM Plan defines the methods and tools used for identifying and controlling the HSCT4.0 project software throughout its development and use. Specifically, it defines the SCM activities, how and when they are to be performed, who is responsible for each activity, and what resources are required. The SCM Plan states that all HSCT4.0/CJOpt software products are to be placed under SCM; in addition to code, these products include makefiles, documentation, test case scripts, and test input and output. The Plan serves as a reference document for the project's SCM procedures. The SCM Plan also includes sections on the schedule for implementing SCM, on the purpose and timing of functional and physical configuration audits, and on Plan maintenance.

Implementing the Plan

A combination of software tools was selected to support the HSCT4.0 SCM activities. An evaluation of commercially available SCM tools was conducted in the early stages of the SCM Plan development.[8] As a result of this evaluation, the consultants recommended the TRUEchange™ [*] product of TRUE Software, Inc.,[9] as the software tool for version control in the HSCT4.0 project. Later, a set of electronic forms, including formal trouble reports, change requests, and promotion notifications, was selected to manage changes to the software. These electronic forms and the associated change-control metrics database had been developed earlier at LaRC and were adapted to the needs of the HSCT4.0/CJOpt project.

[*] The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

Roles and responsibilities. As part of developing the Plan, SCM roles and their corresponding responsibilities were defined. The Plan explicitly identifies the activities required of each role and how these activities are conducted. The SCM roles are the Project Manager, SCM Manager, Software Engineering Manager (SEM), Software Control Board (SCB), developers, testers, users, configuration management staff (CMS), and System Administrator. In the HSCT4.0/CJOpt project, individuals carry out multiple roles. For example, the individual occupying the SEM role also contributes to the software development; therefore this individual acts as SEM, developer, tester, and possibly as a user.

The SCM Manager is the individual responsible for bringing configuration management to the project, while the SEM is responsible for controlling the baselines. The individuals responsible for these two management roles have developed the Plan. In addition, these individuals, along with the Project Manager, permanently reside on the SCB. The SCB organization disposes software change requests and trouble reports. The CMS and system administrator assist the SCM Manager, SEM, and developers in carrying out their activities and in maintaining the software tools.

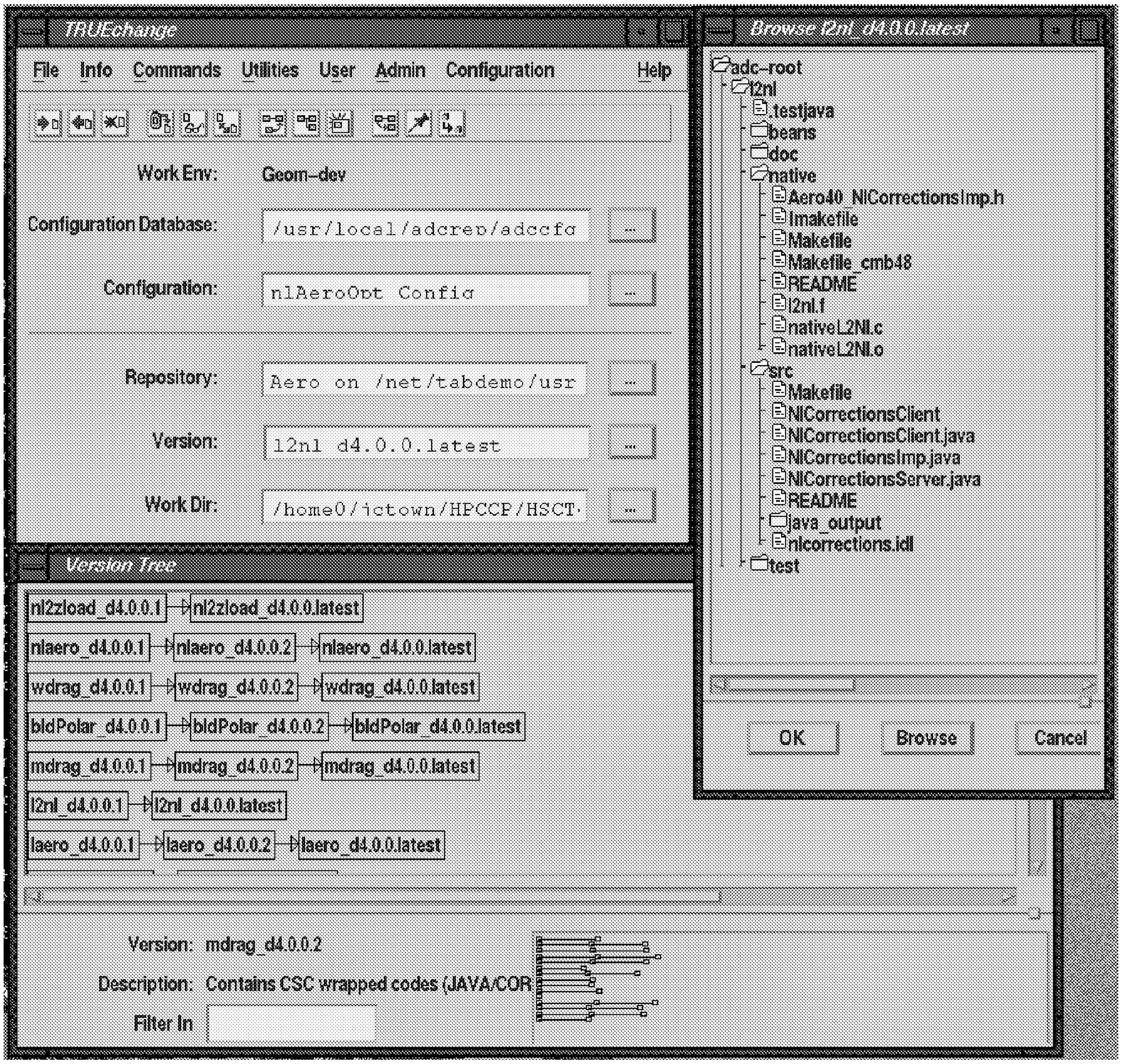
Configuration items. Identifying the *configuration items* (CIs) is a key activity required in developing the Plan. A CI can be defined as an aggregation of software that is treated as a single entity in the configuration management process. Within the TRUEchange tool, the CIs are called *projects*. A TRUEchange project is a group of related files; versions are kept for projects rather than for individual files within projects. For the HSCT4.0 application, projects may contain analysis codes, makefiles, documentation, commercial off-the-shelf (COTS) products, test cases, test data, and research results.

Defining the CI naming and numbering schemes and defining access policies are also CI identification issues that have been addressed in the Plan. TRUEchange project names facilitate identifying projects by including their name phases, releases, minor subreleases, patch subreleases, and version increment numbers. The project name phases that are defined for HSCT4.0/CJOpt are *development*, *test*, and *user*. For example, the name for the project that holds the linear aerodynamics code is "ussaero_d4.0.0.1", where "d" refers to the development phase, "4" refers to the HSCT project major release, there are no minor or patch subreleases, and the internal version increment number is at 1. Figure 3 is a screenshot of some TRUEchange graphical user interface windows, showing project versions within one of the HSCT4.0 repositories.

The TRUEchange tool facilitates defining access privileges to the CIs under SCM. For HSCT4.0, all project members with TRUEchange licenses are allowed to access any of the HSCT4.0 repositories and associated projects. Only the SEM and the CMS have authority to checkpoint (i.e., freeze) a project or to increment a project's phase, release, or subrelease level.

Baselines. An HSCT4.0 product baseline consists of a set of associated CI frozen versions. The baseline is described in TRUEchange as a *configuration*. An advantage of SCM software tools such as TRUEchange is the ability to maintain multiple operational versions of CIs and baselines. Software products developed by the HSCT4.0 team will be baselined at the tester and user levels. The SEM and CMS can easily create and update multiple *reference areas*, each of which contains a software baseline at a distinct development phase. The project team can simultaneously employ these reference areas for independently testing and developing modifications of different software modules.

Procedures. The SCM Plan describes the processes the HSCT4.0 project team will use, after initial development is complete, for setting up the SCM environment, for placing new documents and code into the system, for reporting software problems, for requesting and implementing changes, and for promoting projects to the test and user baseline levels.



Work Env: Geom-dev

Configuration Database: /usr/local/adcrep/adccfg

Configuration: nlAeroOpt Config

Repository: Aero on /net/tabdemo/usr

Version: l2nl d4.0.0.latest

Work Dir: /home0/jctown/HPCCP/HSCT·

```
adc-root
  l2nl
    .testjava
    beans
    doc
    native
      Aero40_NlCorrectionsImp.h
      Imakefile
      Makefile
      Makefile_cmb48
      README
      l2nl.f
      nativeL2Nl.c
      nativeL2Nl.o
    src
      Makefile
      NlCorrectionsClient
      NlCorrectionsClient.java
      NlCorrectionsImp.java
      NlCorrectionsServer.java
      README
      java_output
      nlcorrections.idl
    test
```

```
nl2zload_d4.0.0.1 → nl2zload_d4.0.0.latest
nlaero_d4.0.0.1 → nlaero_d4.0.0.2 → nlaero_d4.0.0.latest
wdrag_d4.0.0.1 → wdrag_d4.0.0.2 → wdrag_d4.0.0.latest
bldPolar_d4.0.0.1 → bldPolar_d4.0.0.2 → bldPolar_d4.0.0.latest
mdrag_d4.0.0.1 → mdrag_d4.0.0.2 → mdrag_d4.0.0.latest
l2nl_d4.0.0.1 → l2nl_d4.0.0.latest
laero_d4.0.0.1 → laero_d4.0.0.2 → laero_d4.0.0.latest
```

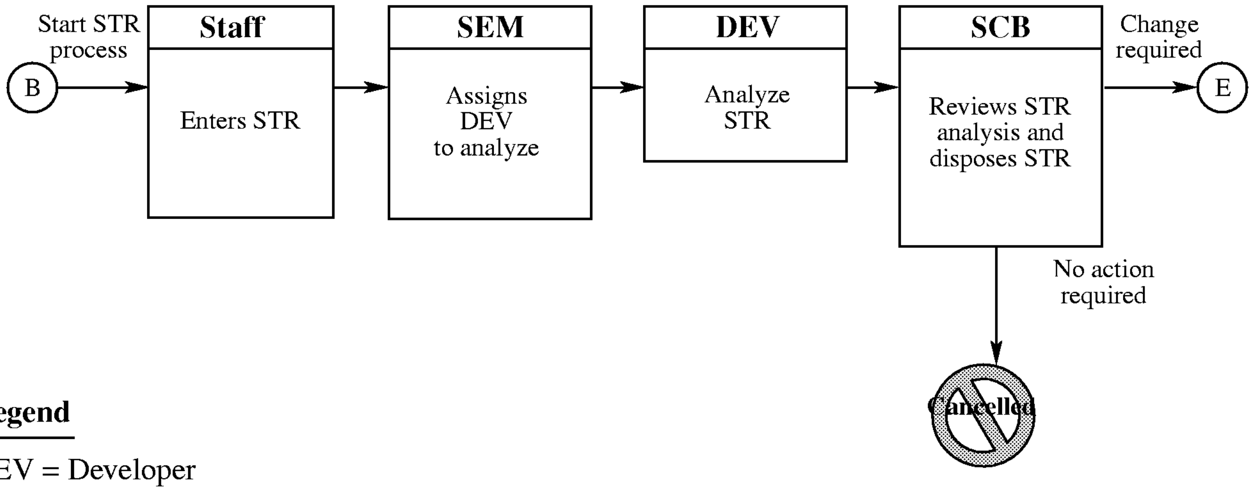
Version: mdrag_d4.0.0.2

Description: Contains CSC wrapped codes (JAVA/COR

Filter In

Figure 3. Screenshot of three TRUEchange graphical user interface windows.

Figure 4 shows, for example, the Plan's Software Trouble Report (STR) process. In this simple process, the SEM assigns a developer to analyze the cause of the reported problem. The developer reports back to the SCB (typically consisting of the SEM, the SCM Manager, the developer, and perhaps the person reporting the trouble) so that the SCB can decide whether the problem warrants making a change to the software. The Software Change Request (SCR) process is similar, but it includes the collection of current SCRs on a candidate list for the SCB; the SCB then sets priorities for the SCR implementation. Changes are implemented by a development process that includes a review by the SEM after the developer has verified the code modification. Promotion processes are used for software that is ready to be promoted to a higher baseline level (i.e., from developer to test or from test to user). These processes include the creation of the appropriate reference area for use by testers (or users) and the notification of all team members of the promotion.



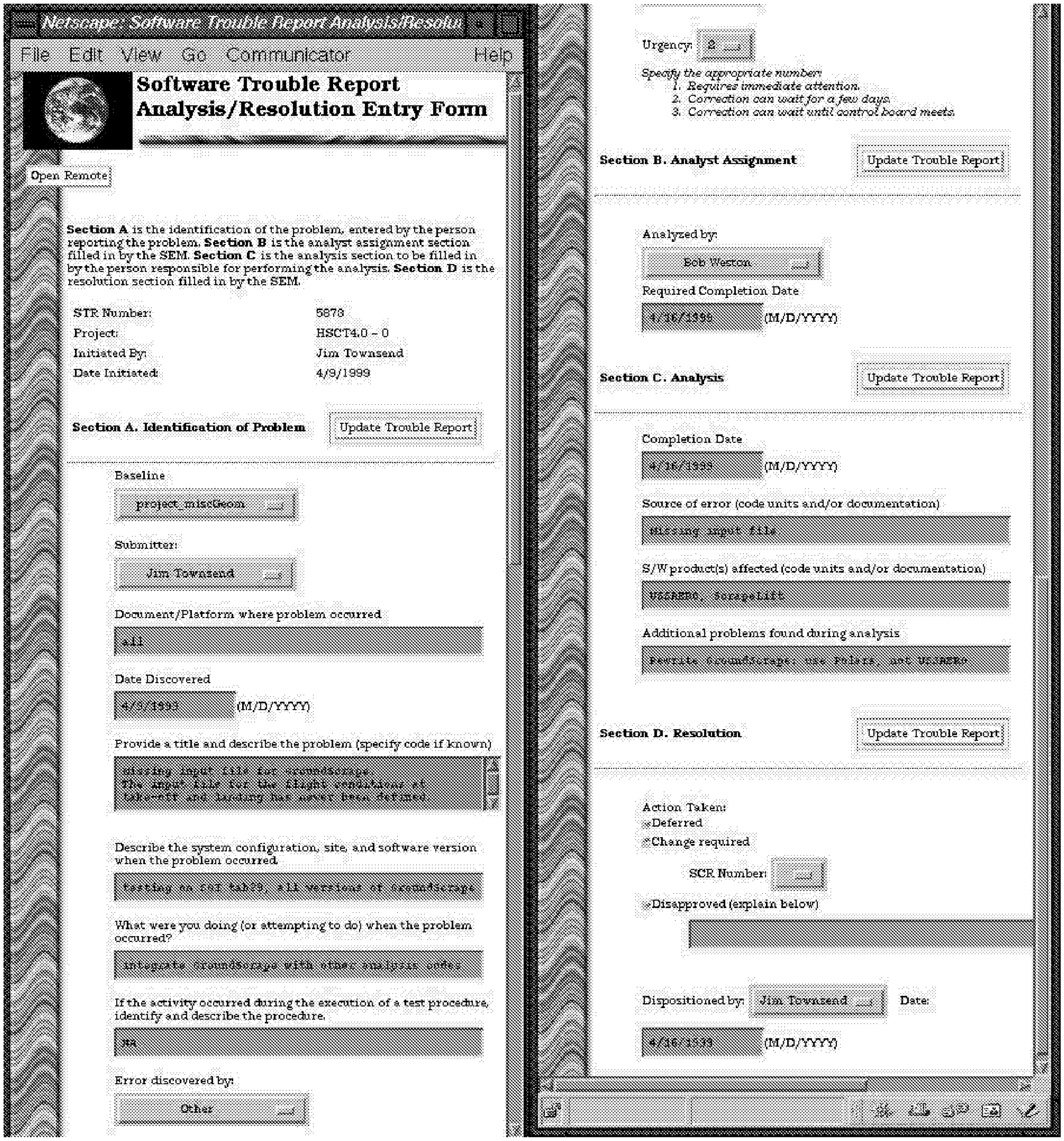
Legend

DEV = Developer
SCB = Software Control Board
SEM = Software Engineering Manager
STR = Software Trouble Report

Figure 4. HSCT4.0/CJOpt Software Trouble Report (STR) process.

Forms and database. The Software Trouble Report (STR), Software Change Request (SCR), and Promotion Notification Form (PNF) are accessible as electronic forms to HSCT4.0/CJOpt team members using on-line Web browsers. Each form has several sections, to be filled out as the process continues. The information is collected in a metrics database, which is automatically updated each time a team member makes entries to an on-line form. As an example, figure 5 shows a filled-out sample of the associated STR on-line form.

The candidate list used by the SCB for prioritization of SCRs is based on the contents of the metrics database. This list is one of several reports that can be generated on request. The others are the STR Report on outstanding problems and the Promotion Notification Form Report on the promotion levels of HSCT4.0/CJOpt baselines. The Project Manager also uses these metrics database reports for accounting purposes.


Netscape: Software Trouble Report Analysis/Resolu
File Edit View Go Communicator Help
Open Remote

Software Trouble Report Analysis/Resolution Entry Form

Section A is the identification of the problem, entered by the person reporting the problem. **Section B** is the analyst assignment section filled in by the SEM. **Section C** is the analysis section to be filled in by the person responsible for performing the analysis. **Section D** is the resolution section filled in by the SEM.

STR Number:	5878
Project:	HSCT4.0 - 0
Initiated By:	Jim Townsend
Date Initiated:	4/9/1999

Section A. Identification of Problem Update Trouble Report

Baseline
project_miscGeom

Submitter:
Jim Townsend

Document/Platform where problem occurred
all

Date Discovered
4/9/1999 (M/D/YYYY)

Provide a title and describe the problem (specify code if known)
Missing input file for GroundScrape
The input file for the flight conditions at take-off and landing has never been defined.

Describe the system configuration, site, and software version when the problem occurred.
testing on cot tab29, all versions of GroundScrape

What were you doing (or attempting to do) when the problem occurred?
integrate GroundScrape with other analysis codes

If the activity occurred during the execution of a test procedure, identify and describe the procedure.
NA

Error discovered by:
Other

Urgency: 2

Specify the appropriate number:
1. *Requires immediate attention.*
2. *Correction can wait for a few days.*
3. *Correction can wait until control board meets.*

Section B. Analyst Assignment Update Trouble Report

Analyzed by:
Bob Weston

Required Completion Date
4/16/1999 (M/D/YYYY)

Section C. Analysis Update Trouble Report

Completion Date
4/16/1999 (M/D/YYYY)

Source of error (code units and/or documentation)
Missing input file.

S/W product(s) affected (code units and/or documentation)
USSAERO, ScrapeLift

Additional problems found during analysis
Rewrite GroundScrape; use Polars, not USSAERO

Section D. Resolution Update Trouble Report

Action Taken:
Deferred
Change required
SCR Number:
Disapproved (explain below)

Dispositioned by: Jim Townsend Date:
4/16/1999 (M/D/YYYY)

Figure 5. Sample Software Trouble Report (STR) on-line form.

CONCLUSIONS

HSCT4.0 Experience with Software Configuration Management

The HSCT4.0/CJOpt project team anticipates that its utilization of SCM will prove to be beneficial. However, even though the tools and processes have been only partially demonstrated, already there have been some problems in the project's SCM implementation.

Pilot project. While the pilot project served to introduce the TRUEchange tool to project members, little else was gained from it for several reasons. The main problem was that the pilot study was scheduled too early in the HSCT4.0 project timeline. The other serious problem was that most of the time allocated towards the pilot study dealt with details of using TRUEchange, rather than with developing clear and complete SCM processes that incorporated the tools. A related problem was that SCM training occurred too early in the project schedule, before the project team members were ready to apply the formal SCM processes. Consequently, refresher training may be needed as processes are put into effect.

Research environment. A research project needs to reach a degree of maturity before its software elements are defined well enough to be managed as configuration items. The HSCT4.0 MDO application has involved a prolonged requirements analysis effort (see figure 2 for a high-level view of the multidisciplinary analysis). The pilot project began too early in the requirements phase and before key decisions had been made involving the framework. Consequently, an application developed with early FIDO concepts was chosen as the pilot problem. During the later stages of the pilot study, after CJOpt had been chosen as the integration method, the consultants set up the SCM structure based on preliminary design concepts. However, because it was early in the design stage, the project team was unable to identify all of the specific configuration items (the SCM projects defining the overall structure within the SCM tool). Subsequently, the SCM structure was modified to make it reflect the design that finally evolved. In retrospect, if the pilot study had been delayed by several months, a small MDO problem containing a subset of the configuration items and employing the CJOpt approach would have been available as the pilot problem. Such a pilot study at the later time would have been much more useful to the project team.

Another problem experienced has been the reluctance of team members to follow the SCM procedures. Team members initially failed to perceive the potential benefits of SCM. Although the team members are recognized experts in their individual disciplines, they are not software engineers. Used to working alone or with only one or two others in their own discipline areas, they are not accustomed to having SCM procedures imposed on them. Reluctance is also due to lack of time to meet milestones and inexperience with the SCM tools. Consequently, there was a tendency to shortcut the process, passing codes between developers directly instead of placing them under configuration management. The result was that often codes, data, and documentation were not placed in the SCM tool according to the planned CI structure, and sometimes unnecessary or duplicate items were stored.

The problems experienced with software development when team members bypassed the SCM system have resulted in a greater acceptance of the need for SCM by the project team. For example, the team has already experienced the inability to regenerate research results consistently and the unintentional use of multiple, inconsistent versions of a code. In one instance, a code being used outside of SCM by one developer was compressed during execution by another developer. As the use of SCM comes to be a habit of the normal project team routine, problems such as these can be avoided.

Tools. The software tools chosen have been well suited for the HSCT4.0/CJOpt project. The concept of version control by project (which keeps changes to related files together) seems naturally applicable to configuration management of software for multidisciplinary optimization. The fact that TRUEchange uses this concept was among the reasons it was recommended for the HSCT4.0/CJOpt project. In order to maintain consistency across its databases, TRUEchange makes renaming and restructuring of projects and repositories very difficult, which inhibited the SCM restructuring. Although not yet fully utilized, the Web-based forms, accessible by Internet browser, have been thoroughly tested and are expected to be convenient. The easy, on-line accessibility of customized forms is considered essential to their acceptance and use. A different software tool for tracking changes had been considered, but it was abandoned because of the difficulty in customizing it to the HSCT4.0 SCM process and because it was not easily accessible in the Unix environment used for HSCT4.0/CJOpt development and testing.

Lessons Learned

Some of the lessons that have been learned from the experience so far with applying software configuration management to the HSCT4.0/CJOpt multidisciplinary optimization project are listed below:

- The SCM Plan and its implementers need to be specific in defining the procedures and responsibilities; it is helpful to provide templates or examples of what is expected.
- It is important to allow adequate time in the schedule to introduce SCM and to perform it.
- The project plan and schedule and all subcontractor tasks must explicitly address the use of SCM.
- The team members must accept SCM, preferably by understanding its value and potential benefits (i.e., willingly, rather than by coercion).
- It is not enough for team members to place their final products into an SCM repository; they must actively and consistently apply SCM in order to reap the benefits.

REFERENCES

1. Sobieszczanski-Sobieski, J., and Haftka, R. T., "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments," AIAA Paper 96-0711, Jan. 1996.

2. Weston, R. P., Townsend, J. C., Eidson, T. M., and Gates, R. L., "A Distributed Computing Environment for Multidisciplinary Design," *5th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, AIAA-94-4372-CP, Sept. 1994, pp. 1091-1097.

3. Vinoski, S., "CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments," *IEEE Communications*, Vol. 14, No. 2, 1997, pp. 46-55.

4. Sistla, R., Dovi, A. R., Su, P., and Shanmugasundaram, R., "Aircraft Design Problem Implementation under the Common Object Request Broker Architecture," AIAA Paper 99-1348, March 1999.

5. Arthur, James; Frakes, William; Gupta, Sanjay; Cannon, Martha; Groener, Markus; and Khan, Zakia: Report on Quasi-Experiment for Evaluating SEES. Dept. of Computer Science, Virginia Tech, January 15, 1996.

6. Software Engineering Process Guidebook, Software Configuration Management Planning Volume, NASA LaRC, Software Engineering and Analysis Lab, Dec. 1995.

7. *IEEE Standard for Software Configuration Management Plans*, ANSI/IEEE Std 828-1998.

8. Computer Sciences Corporation, *Commercial-Off- the-Shelf Software Configuration Management Evaluation Report for NASA Langley Research Center*, Contract No. NAS1-20431, Jan. 1998.

9. *TRUEchange Command Line User Interface*, *Version 2.1*, TRUE Software, Inc., Waltham, MA, 1997.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)	**2. REPORT DATE** June 1999	**3. REPORT TYPE AND DATES COVERED** Technical Memorandum

4. TITLE AND SUBTITLE Configuration Management of an Optimization Application in a Research Environment	**5. FUNDING NUMBERS** 509-10-11-01
6. AUTHOR(S) James C. Townsend, Andrea O. Salas, and M. Patricia Schuler	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA, Langley Research Center Hampton, VA 12681-2199	**8. PERFORMING ORGANIZATION REPORT NUMBER** L-17868
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** NASA/TM-1999-209335

11. SUPPLEMENTARY NOTES

Based on presentation to Engineering Foundation Conference on Optimization in Industry, June 6-11, 1999, Banff, Alberta, Canada

12a. DISTRIBUTION / AVAILABILITY STATEMENT Unclassified-Unlimited Subject Category : 61 Availabilty: NASA CASI (301) 621-0390 Distribution: Nonstandard	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

Multidisciplinary design optimization (MDO) research aims to increase interdisciplinary communication and reduce design cycle time by combining system analyses (simulations) with design space search and decision making. The High Perfornance Computing and Communication Program's current High Speed Civil Transport application, HSCT4.0, at NASA Langley Research Center involves a highly complex analysis process with high-fidelity analyses that are more realistic than previous efforts at the Center. The multidisciplinary processes have been integrated to form a distributed application by using the Java language and Common Object Request Broker Architecture (CORBA) software techniques. HSCT4.0 is a research project in which both the application problm and the implementation strategy have evolved as the MDO and integration issues became better understood. Whereas earlier versions of the application and integrated system were developed with a simple, manual software configuration management (SCM) process, it was evident that this larger project required a more formal SCM procedure. This report briefly describes the HSCT4.0 analysis and its CORBA implementation and then discusses some SCM concepts and their application to this project. In anticipation that SCM will prove beneficial for other large research projects, the report concludes with some lessons learned in overcoming SCM implementation problems for HSCT4.0.

14. SUBJECT TERMS Software Configuration Management Multidisciplinary Optimization Frameworks Computer-Aided Design	**15. NUMBER OF PAGES** 17
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT**